INF110 Ukeoppgaver: Uke 6

OPPGAVE 1

Regnestykket 1.23 + ((4.56 * 7.89) / (3.3 + 4.44)) kan skrives på postfiks form uten bruk av parenteser slik:

```
1.23 4.56 7.89 * 3.33 4.44 + / +
```

Når uttrykk på postfiks form skal beregnes er det naturlig å bruke en stakk av tall. Målet er da at svaret står alene igjen på stakken når beregningen er ferdig.

a) Forklar først med ord hva slags stakk-operasjoner som må gjøres når man leser inn et postfiks uttrykk sekvensielt, og får inn henholdsvis et tall eller en av de fire operasjonene. Tenk deg også en operasjon som henter ut svaret til slutt, slik at det blir i alt seks operasjoner.

b) Anta at man har en standard-pakke for stakker av reelle tall, og bruk denne til å implementere de seks operasjonene.

c) Lag en direkte implementasjon av de seks operasjonene som bruker en array. Legg også inn i denne modulen mekanismer som tester om uttrykket er et korrekt uttrykk.

OPPGAVE 2

Vi skal bygge opp et binært tre, der nodene er av klassen:

```
public class Node {
  int data;
  Node venstre, hoyre;

  public Node(int d) {
       data = d;
  }
}
```

Treet er gitt som en sekvens av tall, der hver node er beskrevet med tre tall. Det første er nodens data, og de to andre er egentlig boolske verdier (der 1=true og 0=false) som angir om noden har et venstre og/eller høyre ikketomt subtre. Nodene kommer i prefiks rekkefølge, og er kodet som en ubrutt strøm av tall. Tegn eksempler på trær, og hvordan de vil se ut som tall-sekvenser.

Vi skal lese disse dataene, og bygge opp treet etter hvert. Roten av treet skal angis av en Node-variabel "rot". Du kan anta at det finnes en metode "int nesteTall()" som gir det neste tallet i sekvensen.

Oppgaven skal løses på to måter:

a) Bruk en rekursiv metode.

b) Bruk en stakk som inneholder (peker til) de nodene som forventer, men som ennå ikke har fått, et høyre subtre.

c) Til slutt kan man se på tilfellet at treet stammer fra et uttrykk med bare vanlige binære operasjoner. Forklar at om man legger et slikt tre ut i prefiks rekkefølge, så trenger man IKKE de boolske merkene for å kunne rekonstruere treet. Du kan tenke deg at operandene i uttrykket bare er positive heltall, og at de fire binære operasjonene er representert som -1='+', -2='-', -3='*' og -4='/'. Dermed kan prefiks-utgaven av uttrykkstreet legges ut som en tall-sekvens med ett tall for hver node. Se gjennom a) og b) over, og angi hvilke forenklinger man kan gjøre.

d) Lag til slutt en rekursiv metode som beregner verdien av et uttrykk lagt ut i prefiks format, i stedet for å bygge opp treet.

OPPGAVE 3

Anta at vi i et program skal arbeide med "mengder av heltall", og at vi velger å representere disse mengdene i arrayer, slik at tallene ligger SORTERT fra starten av arrayene, og slik at hvert tall bare forekommer EN gang. Til hver array må også høre et antall, slik at hver mengde f.eks. er representert ved et objekt av klassen:

```
class Mengde {
  int[] m = new int[1000];
    int ant;
}
```

a) Lag en metode "void union(Mengde a, Mengde b, Mengde c)" som tar unionen av innholdet av a og b, og setter det inn som (nytt) innhold i c. Metoden skal bare gå EN gang gjennom hver mengde, og skal foreta en slags "fletting". Lag også en tilsvarende metode "snitt".

b) Vi skal på disse mengdene ha følgende operasjoner (i tillegg til snitt og union som under punkt a):
   - Sett et gitt tall 'i' inn i en mengde, om det ikke er der fra før av.
   - Fjern et gitt tall 'i' fra en mengde, dersom det finnes i mengden.
   - Er tallet 'i' med i mengden?

Planlegg en implementasjon av disse operasjonene, og angi hvilken orden de blir av. Kanskje blir ordenen avhengig av hvilke parametre operasjonen får?

OPPGAVE 4

Vi har gitt et (ordnet, men ikke binært) tre, der subnodene til hver node ligger i en vanlig enkelt-lenket liste. Nodene er av følgende klasse:

```
class Node {
  String navn;
  Node forsteBarn, nesteSosken;
  int dybde, hoyde, antall, bladAvstand;
  // eventuelle tillegg
}
```

Vi definerer "bladAvstanden" til en node som veilengden ned til nærmeste blad. Høyden til en node er tilsvarende veilengden ned til fjerneste blad. Skriv en rekursiv metode som går gjennom treet, og som setter inn de riktige verdiene i alle nodene. Attributtet "antall" skal angi antall noder i subtreet definert av noden (inklusive den selv). De to Node-variablene i nodene er altså satt på forhånd, og skal ikke forandres.

OPPGAVE 5

For å kunne vurdere hvor skjeve binære søketrær med n noder blir i gjennomsnitt, vil vi generere alle permutasjoner av tallene fra 0 til n-1. For hver permutasjon bygges et søketre der verdiene ankommer i rekkefølgen gitt av permutasjonen. For hvert tre beregnes høyde H (maksimal avstand fra rot til blad) og gjennomsnittlig søkelengde GS (gjennomsnittlig avstand fra roten til hver av nodene i treet).

De tallene vi er interessert i er gjennomsnittet av H og GS, tatt over de n! trærne vi får ut fra alle permutasjonene av lengde n. Skriv et program som finner disse tallene for en gitt verdi av n.